CLAIMS

What is claimed is:

1. A method comprising the steps of:

a) identifying at least one statement within a synthesizable source code; and

b) synthesizing the source code into a gate-level netlist including at least one instrumentation signal, wherein the instrumentation signal is indicative of an execution status of the at least one statement.

2. The method of claim 1 wherein step b) includes the step of:

i) generating instrumentation logic to provide the instrumentation signal as if the source code included a corresponding signal assignment statement within a same executable branch of the source code as the identified statement.

3. The method of claim 1 wherein step b) includes the steps of:

i) initializing a marker to a first value at the beginning of a process within the source code; and

ii) setting the marker to a second value within a same executable branch of the source code as the identified statement.

4. The method of claim 3 further comprising the step of:

iii) assigning the value of the marker to the instrumentation signal at the end of the process.

5. A method of generating a gate level design, comprising the steps of:

a) creating an instrumentation signal associated with at least one synthesizable statement contained in a source code; and

b) synthesizing the source code into a gate-level design having the instrumentation signal.

6. The method of claim 5 wherein step a) further comprises the step of:

i) inserting a unique variable assignment statement into the source code, wherein the variable assignment statement is adjacent to at least one associated sequential statement; and

ii) inserting a unique output signal assignment statement into the source code, wherein the unique output signal is assigned a value associated with the unique variable.

7. The method of claim 6 wherein the variable is assigned a first value in step a)i), the method further comprising the step of:

iii) modifying the source code to initialize the unique variable to a second value.

8. The method of claim 5 wherein step a) is repeated to create a unique instrumented output signal for each list of sequential statements in the source code, wherein each list corresponds to a synthesizable executable branch of the source code.

9. The method of claim 5 further comprising the step of:

c) generating cross-reference instrumentation data mapping each statement in a selected list to the instrumented output signal associated with that list for every list in the source code.

10. The method of claim 9 further comprising the steps of:

d) simulating the gate level design using at least one of the instrumentation signals to establish a simulation breakpoint.

11. The method of claim 5 further comprising the steps of:

c) displaying the source code, wherein at least one statement within a selected list is highlighted if the instrumentation signal corresponding to the selected list changes to a pre-determined value.

12. A method of generating a gate-level netlist, comprising the steps of:

a) receiving source code including synthesizable statements;

b) inserting a unique local variable assignment statement into the source code for each branch of code having a list of at least one sequential statement, wherein the unique local variable assignment statement is adjacent to at least one statement within the list;

c) inserting a corresponding instrumentation signal assignment statement into the source code for each of the inserted local variables, wherein the instrumentation signal is assigned a value of the corresponding unique local variable; and

d) synthesizing the source code into a gate-level design including the instrumentation signals.

13. The method of claim 12 wherein step b) further comprises the steps of:

i) assigning each unique local variable a first value; and

ii) initializing each local variable with second value.

14. The method of claim 12 further comprising the step of

e) mapping every statement within a selected list to the corresponding instrumentation signal for that selected list as cross-reference instrumentation data.

15. The method of claim 12 further comprising the steps of:

e) setting a breakpoint at a selected statement of the source code;

f) identifying the instrumentation signal corresponding to the list associated with the selected statement as a breakpoint signal; and

g) simulating the gate-level design, wherein simulation is halted at a simulation cycle that results in the breakpoint signal transitioning to a pre-determined value.

16. A method of generating a gate level netlist, comprising the steps of:

a) receiving source code including synthesizable statements;

b) modifying the source code to generate a corresponding sampled version of each signal event in a selected process;

c) modifying the source code to duplicate the selected process;

d) replacing each occurrence of a selected signal event with the corresponding sampled version in the duplicated process;

e) replacing each list of sequential statements within an executable branch of the duplicated process with a unique variable assignment statement;

f) modifying the duplicated process to include an instrumentation signal assignment for each unique variable; and

g) synthesizing the modified source code into a gate-level design.

17. The method of claim 16 wherein step e) further comprises the steps of:

i) assigning the unique variables a first value; and

ii) initializing the unique variables with second value.

18. The method of claim 16 further comprising the step of

e) mapping every statement within each selected list to its corresponding instrumentation signal.

19. The method of claim 16 further comprising the steps of:

h) setting a breakpoint at a selected statement of the source code;

i) identifying the instrumentation signal corresponding to the list associated with the selected statement as a breakpoint signal; and

j) simulating the gate-level design, wherein simulation is halted at a simulation cycle that results in a transition of the breakpoint signal to a pre-determined value.

20. A method of debugging a gate-level design including the steps of:

a) setting a breakpoint at a selected statement of a synthesizable source code;

b) inserting a local variable assignment statement adjacent to at least one statement in a list of sequential statements , wherein the list corresponds to an executable branch of the source code including the selected statement;

c) modifying the source code to include an instrumentation signal assignment statement for the local variable; and

d) generating a gate-level design from the modified source code.

21. The method of claim 20 further comprising the steps of:

e) simulating the gate-level design, wherein simulation is halted at a simulation cycle that results in a transition of the instrumentation signal to a pre-determined value.

22. The method of claim 20 wherein step b) further comprises the steps of:

i) assigning the local variable a first value; and

ii) initializing the local variable with second value.

23. The method of claim 20 further comprising the step of

e) mapping every statement within the executable branch of source code to the instrumentation signal.

24. A method of simulating a gate-level design comprising the steps of:

a) identifying a sensitivity list of a process;

b) generating logic to identify signal events for any signal in the sensitivity list; and

c) identifying the process as active during simulation when a signal event occurs for any signal in the sensitivity list.

25. The method of claim 24 wherein step c) further comprises the step of:

i) highlighting a source code description of the process displayed during simulation.

26. The method of claim 24 wherein step b) further comprises the step of:

i) sampling each signal in the sensitivity list to generate corresponding instrumented signals; and

ii) comparing each signal in the sensitivity list with its corresponding instrumented signal to test each signal in the sensitivity list for an event.

27. The method of claim 26 wherein step c) further comprises the step of:

i) generating an active process output signal defined by logically ORing the results of the comparisons.

28. A storage medium having stored therein processor executable instructions for generating a gate-level design from a register transfer level

(RTL) synthesizable source code, wherein when executed the instructions enable the processor to synthesize the source code into a gate-level netlist including at least one instrumentation signal, wherein the instrumentation signal is indicative of an execution status of at least one synthesizable statement of the source code.

29. The storage medium of claim 28 wherein the processor performs the steps of:

i) inserting a unique variable assignment statement into the source code, wherein the variable assignment statement is adjacent to at least one associated sequential statement; and

ii) inserting a unique output signal assignment statement into the source code, wherein the unique output signal is assigned a value associated with the unique variable.

30. A storage medium having stored therein processor executable instructions for generating a gate-level design from a register transfer level (RTL) synthesizable source code, wherein when executed the instructions enable the processor to perform the steps of:

a) inserting a unique local variable assignment statement into the source code for each branch of code having a list of at least one sequential statement, wherein the unique local variable assignment statement is adjacent to at least one statement within the list;

b) inserting a corresponding instrumentation signal assignment statement into the source code for each of the inserted local variables,

wherein the instrumentation signal is assigned a value of the corresponding unique local variable; and

c) synthesizing the source code into a gate-level design including the instrumentation signals.

31. The storage medium of claim 30 having stored therein further instructions to enable the processor to perform the step of:

d) mapping every statement within each selected list to its corresponding instrumentation signal.

32. A storage medium having stored therein processor executable instructions for debugging a gate level design during simulation, wherein when a breakpoint is set at a selected statement of a register transfer level (RTL) synthesizable source code the instructions enable the processor to perform the steps of:

a) inserting a local variable assignment statement adjacent to at least one statement in a list of sequential statements within the source code, wherein the list corresponds to an executable branch of the source code including the selected statement;

b) modifying the source code to include an instrumentation output signal assignment statement for the local variable; and

c) generating a gate-level design from the modified source code.

33. The storage medium of claim 32 having stored therein further instructions to enable the processor to perform the step of:

d) mapping every statement within each selected list to its corresponding instrumentation signal.